Institute for Mathematical Studies
in the Social Sciences
Stanford University
Stanford, California
December 31, 1968

Reference: NASA Research Grant NGR-05-020-244

Status Report (July 1, 1968 - December 31, 1968)

The major effort of the past two months has been the design of a computer-assisted instruction system for teaching programming languages. One of the first programming languages which will be taught using this system will be AID.* Although the immediate goal is to develop a self-contained course in AID programming, the secondary goal, which is, perhaps, of greater importance, is to produce an instructional system which will be equally well suited to teaching any interactive programming language, such as BASIC, time-shared FORTRAN, time-shared ALGOL or APL.

Except for detailed course outlines and a few introductory lessons, the major effort has been not in curriculum construction but in system design, although the two influence one another to such a degree that it is not feasible to consider them independently. In the following remarks about the system design frequent reference will be made to the AID curriculum, but the reader may substitute "BASIC," for example, for "AID" almost everywhere without distorting the meaning.

---

*Algebraic Interpretive Dialogue is an interactive programming language based on JOSS and implemented by Digital Equipment Corporation for PDP computers. Both the language and the course were described in some detail in previous progress reports.

The view of the designer has been that a course in programming is generally aimed at relatively mature students, high school or college age students, who are capable of (and interested in) making decisions about the course of study. For this reason the major part of the system will be designed to allow the student as much control over the sequence of lessons and problems as is feasible. ("Feasible" is used here to mean "feasible from a system designer's point of view." Practical considerations such as response time may override other considerations in some instances.) Thus, in the lesson strand (core curriculum), the student will be allowed to skip from lesson to lesson in any order, and to skip any problems within a lesson, in much the same way that he might read a textbook, skipping certain chapters, skimming others, and studying some in great detail. The student may, of course, be advised and guided in these decisions. For example, there will be in the AID course lessons which are self-tests, the primary purpose being to provide the student with an evaluation of his progress, and to offer advice based on his test score.

The lesson strand will be essentially linear, much as a textbook, with the assumption that most students will begin with Lesson 1 and continue straightforwardly through the sequence. Each problem will be displayed in turn and the student's response will be immediately evaluated; reinforcement will be made by means of messages such as "Very good," or "Wrong. Try again." A correct response will cause a branch to the next problem in sequence but the system will wait indefinitely for a student who is making a sequence of incorrect responses. (Of course, the student has it within his power to skip the problem at any time he chooses.) The curriculum writer will supply "hints" for the problems in the

strand; these hints will be available to the student upon demand; if the writer fails to supply a hint, the default condition in the system will cause a message such as "No hint was written for this problem" to be sent to any student who requests a hint.

Thus the lesson strand including self-tests could serve as a complete course for highly-motivated students, for example, an engineer who wishes to learn as quickly and efficiently as possible as much of algebraic programming language as needed to do a job at hand.

In an organized course, in a high school or training school, for example, it is not always desirable to have the student in complete control; it may also be necessary to have some means of external evaluation of student progress (Grades must be given. Concrete goals must be imposed on students who are not self-motivated. Etc.). Homework assignments and tests are the two most commonly used methods for evaluating individual students. In order to provide the teacher with such tools for evaluation, the system will allow for exercise strands and a test strand in addition to the lesson strand. Both exercise and test strands are viewed by the system designer as tools to be controlled by the teacher, in contrast to the lesson strand which is a tool to be controlled by the student. Tests, for example, can be taken only at certain times, as decided by the teacher, and in the test strand, unlike the teaching strand, the student will not be immediately informed of the evaluation of his response, but such information will be collected and later released to the teacher. Within a test, the student will be allowed to skip items and return to them later during the same period; he will also be allowed to delete responses and enter replacements, again with the restriction that it be during the same period. Furthermore, since tests in

programming courses are ordinarily "open book" tests, the student will be allowed to set the test aside while he looks at any lesson material and then later return to the test.

The exercise strands (in the AID course there will be two exercise strands, one for homework assignments and one for extra-credit problems) will also be controlled by the teacher, at least to the extent that individual lessons will be available to the student only as decreed by the teacher, thus allowing him to enforce a schedule for the submission of outside work. There will be more student interaction in the exercise strands than in the test strand, however, since the student will be immediately informed of the evaluation of his responses. Results of the evaluation will also be stored to be delivered to the teacher later.

The teacher will be free to exercise no control over tests and exercises if he desires; that is, he may simply make all tests and exercises available to all students at all times.

In summary, the teaching course will consist of several strands, which are logically (but, it is hoped, not contextually) independent. The strands will be of three types: lessons, tests, and exercises. The lesson strands will be almost entirely under the control of the student and will supply him with an immediate evaluation of his responses; in the first version of the AID course there will be only one lesson strand although it is conceivable that several lesson strands might be desirable for other courses, for instance, one for a main stream of lessons in the use of PL/1, a second for the engineering applications of PL/1, and a third for the business applications. The second kind of strand, the test strand, will be largely controlled by the teacher and will not

give feedback to the student. The third kind of strand, the exercise strands, is a combination of the first two kinds; both the student and teacher receive evaluation but the lessons are available only when requested by the teacher.

In order to make such a teaching system applicable to a variety of programming languages, there are a number of practical decisions which must be made by the curriculum designer since the requirements will vary from one course to the next, and cannot be made in advance of knowledge of the specific programming language to be taught. These decisions include such matters as the allowable maximum length of the student response. In some cases, such as in the AID course, it is desirable to limit student responses to one line of print since AID itself does so, whereas it would be difficult to teach a freely formatted and cumbersome language such as LISP with such a restriction. Because of programming considerations an upper bound on the allowable length must be set by the system designer but the maximum for a particular application should be set by the curriculum designer. Other decisions which should be left to the curriculum designer include such things as the specification of the character which will be used to erase the preceding single character, the specification of the "erase-the-last-line" character and the "erase-the-entire-response" character. The character which will be used by the student to signal the end of his answer should also be decided by the curriculum designer. Whether or not it is feasible to leave all of these decisions to the writer is now being considered.

The teaching system should also be flexible enough to allow for the addition of new procedures for analysing student responses. There will be standard analysis routines available to all writers but it is impossible to anticipate the needs

for a course in a programming language which has not yet been invented! The standard analysis routines will include routines applicable to multiple-choice problems with a single correct answer, multiple-choice problems with several correct answers, and simple constructed response problems for which the correct response must be an exact match with the coded standard. There will also be routines which will de simple syntactic analyses such as determining whether or not a given string of characters is an integer, or a non-negative integer, or a real number in scientific notation, etc. One routine will check for numeric equivalence within specified tolerances, another will check for the use of specified keywords, while still another will check to find out if a specified keyword was not used.

The detailed specifications and development of analysis routines has been a large part of the work of the past two months, and will continue to be a major effort in the near future since the heart of any system of computer-assisted instruction lies in such routines. The procedures for analysing student responses must be reliable, fast and discriminatory. The need for reliability needs no justification. The justification for requiring speed is that the system response time, which is so critical to the success of computer-assisted instruction, is largely dependent upon the time taken by the analysis procedures. Although it is probable that older students are more tolerant of delays than, say, elementary school children, (and it may even be argued that students of programming might as well learn early to tolerate the idiosyncracies of computers) it is felt that response times of 10 seconds would be cause for severely limiting the number of simultaneous users of the system.

Analysis procedures should be discriminatory in the sense that they distinguish, insofar as is practical, not only between correct and incorrect responses but also between different kinds of incorrect responses. It is only in this way that pertinent diagnostic messages can be given to students who respond incorrectly. Since there is evidence to show that the overwhelming preponderance of incorrect responses fall into a very small number of classes (regardless of subject matter), most analysis procedures need discriminate between only two, or perhaps three, kinds of incorrect responses in order to be quite effective.

Preliminary specifications (always subject to modification after the actual job of programming starts) have been written for all of the analysis procedures mentioned above, and the procedures themselves have been written for four of the simpler analyses.

Another large part of the work in the past few weeks has been the development of a preliminary version of the coding language, which should be simple to learn and use, and should allow the coder or curriculum writer to be as taciturn as he wishes without denying him any of the flexibility inherent in the teaching system. Several lessons have been coded in the preliminary coding language and the final version of the coding language will be completed within the next few weeks.

The teaching system, in its entirety, will consist of the teaching strands mentioned above, including the routines for displaying problem text and receiving student responses, the analysis procedures, the action procedures (that is, the routines for sending diagnostic messages to the students, giving hints, etc.), the routines for interpreting code written in the coding language, and a supervisor program to allow student-controlled branching between strands. There will

also be proctor routines to allow for teacher control of certain strands, a time-sharing monitor to allow simultaneous use of the system by many users, and routines for data collection.

In the next six months the teaching system will be developed sufficiently to allow a small scale pilot study of the lesson strand. Curriculum material for the introductory section of the lesson strand of the AID course will be developed and tested on a few students; it is hoped that the results of the pilot study will lead to meaningful revisions of both the teaching system and the curriculum material.

Report submitted by

Richard C. Atkinson
Professor of Psychology